Unconstrained Optimization and Neural Networks

Jinning Li
lijinning@sjtu.edu.cn
jinningli.cn

Introduction

- Single Neuron
- Feedforward Neural network
- A **map** from $\mathbb{R}^n$ to $\mathbb{R}^m$, **n**: number of input, **m**: number of output.
- The information about the mapping is "stored" in the weights over all the neurons
- Training set:
  $$\{(\boldsymbol{x}_{d,1}, \boldsymbol{y}_{d,1}), \ldots, (\boldsymbol{x}_{d,p}, \boldsymbol{y}_{d,p})\}$$
- Function approximators:
  $$\boldsymbol{y}_{d,i} = \boldsymbol{F}(\boldsymbol{x}_{d,i})$$

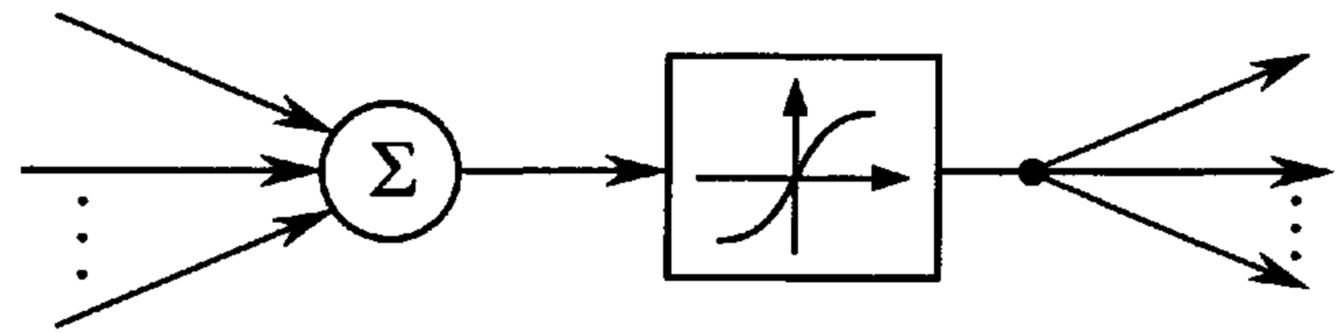


Figure 13.1 A single neuron

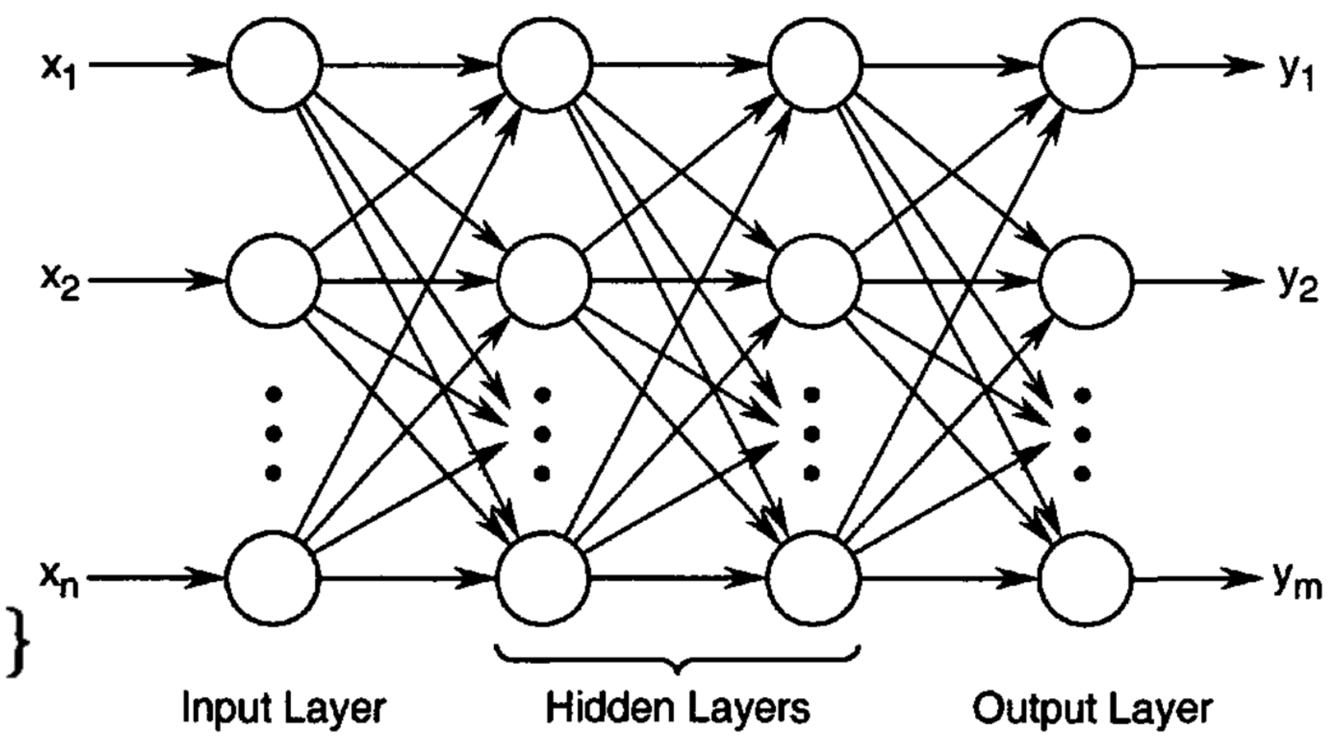


Figure 13.3 Structure of a feedforward neural network

SINGLE-NEURON TRAINING

- Activation function: Identity (linear function with unit slope)

$$y = \sum_{i=1}^{n} w_i x_i = \boldsymbol{x}^T \boldsymbol{w}$$

- Training dataset:

$$\begin{aligned} \boldsymbol{X}_d &= [\boldsymbol{x}_{d,1} \cdots \boldsymbol{x}_{d,p}] \\ \boldsymbol{y}_d &= \begin{bmatrix} y_{d,1} \\ \vdots \\ y_{d,p} \end{bmatrix}. \end{aligned}$$

- Objective function:

$$\text{minimize } \frac{1}{2} \sum_{i=1}^{p} \left(y_{d,i} - \boldsymbol{x}_{d,i}^T \boldsymbol{w} \right)^2$$

- Solution:
  - Case 1: $p \leq n$
  - Case 2: $p > n$

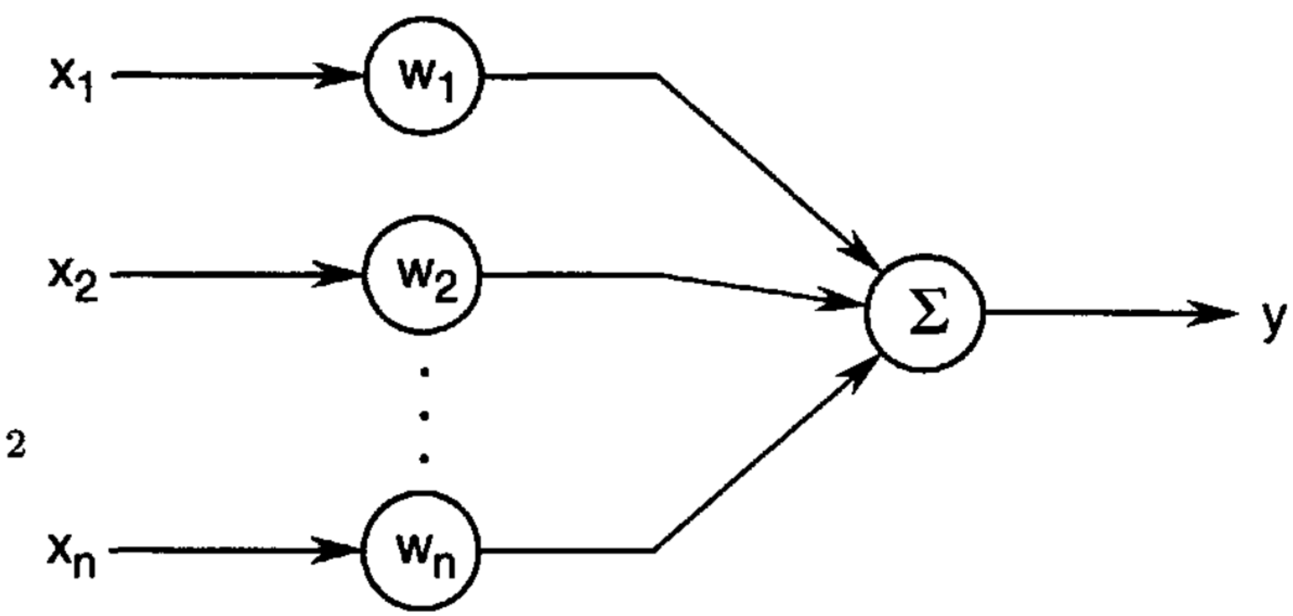


Figure 13.4 A single linear neuron

Case 1: p≤n

- Assume $\operatorname{rank} \boldsymbol{X}_d^T = p$, infinite solution to $\boldsymbol{y}_d = \boldsymbol{X}_d^T \boldsymbol{w}$
- Task: $$\begin{aligned}&\text{minimize } \|w\| \\ &\text{subject to } X^T w = y\end{aligned}$$
- 其实是个最小二乘问题，上回书说到：

定理 12.1 能够最小化 $\|\boldsymbol{Ax}-\boldsymbol{b}\|^2$ 的向量 $\boldsymbol{x}^*$ 具有唯一性，可通过求解方程组 $\boldsymbol{A}^\top\boldsymbol{Ax} = \boldsymbol{A}^\top\boldsymbol{b}$ 得到，即 $\boldsymbol{x}^* = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b}$。 □

- An iteration method without calculate inverse: Kaczmarz's algorithm:

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \mu \frac{e_k \boldsymbol{x}_{d,R(k)}}{\|\boldsymbol{x}_{d,R(k)}\|^2},$$

where $\boldsymbol{w}^{(0)} = \mathbf{0}$, and

$$e_k = y_{d,R(k)} - \boldsymbol{x}_{d,R(k)}^T \boldsymbol{w}^{(k)}.$$

Widrow and Hoff

- Adaline (Adaptive Linear System)

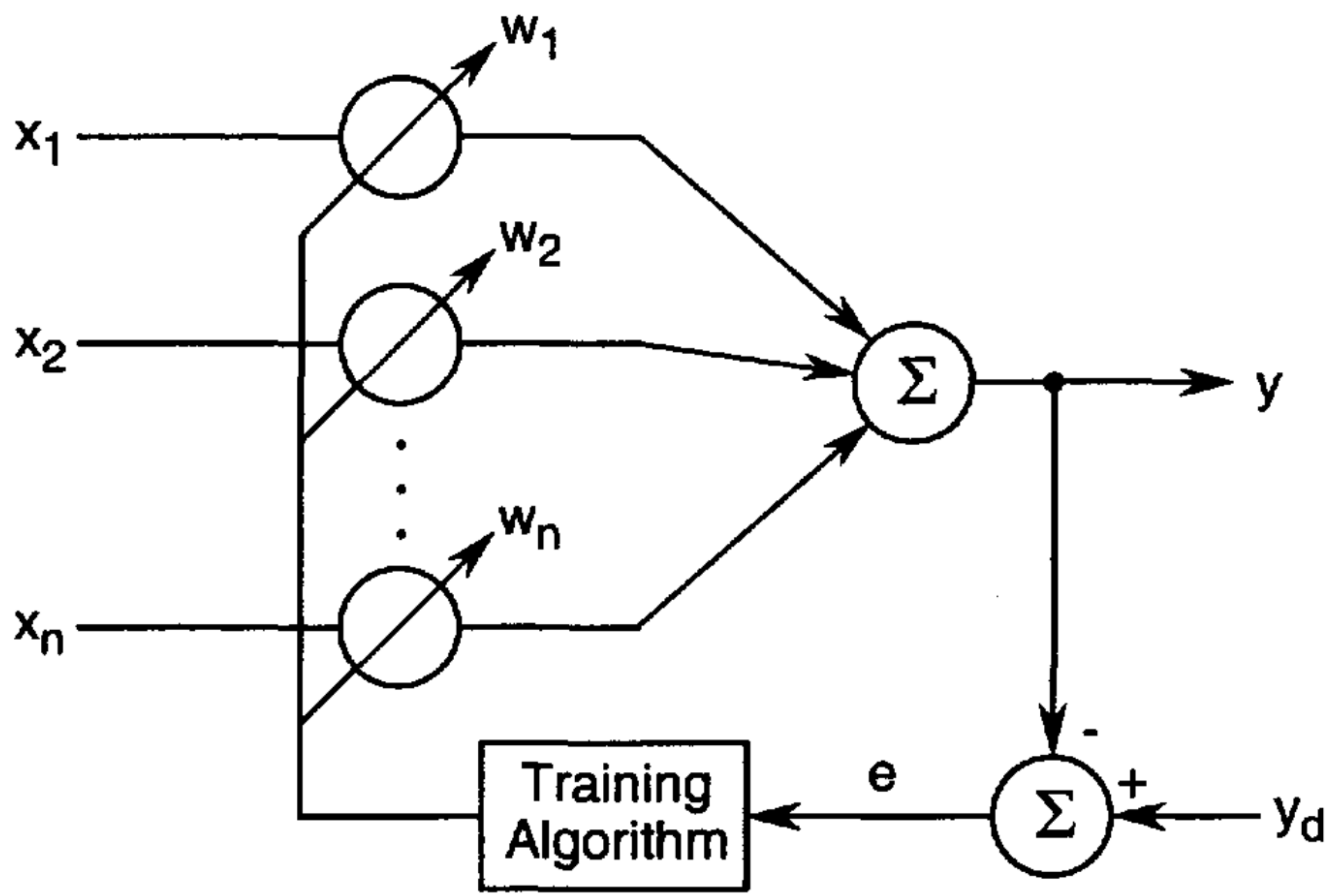


Figure 13.5 Adaline

Case 2: p>n

- More training points than the number of weights
- $\frac{1}{2}\|\boldsymbol{y}_d - \boldsymbol{X}_d^T \boldsymbol{w}\|^2$: strictly convex quadratic function of *w*, because $\boldsymbol{X}_d \boldsymbol{X}_d^T$ is a positive definite matrix.
- Unconstrained optimization algorithms: gradient descent

$$\text{minimize} \quad \frac{1}{2}||y - X^T w||^2$$

$$f(w) = \frac{1}{2}||y - X^T w||^2$$

$$= \frac{1}{2}\left(w^T X\right)\left(X^T w\right) - y^T X^T w + \frac{1}{2} y^T y$$

$$= \frac{1}{2}\left(w^T X X^T w\right) - (Xy)^T w + \frac{1}{2} y^T y$$

- Objective function:
- Derivative:

$$Df(w) = XX^T w - Xy$$

Case 2: p>n

- Iteration formula: $$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \alpha_k \boldsymbol{X}_d \boldsymbol{e}^{(k)}$$

$$\text{where } \boldsymbol{e}^{(k)} = \boldsymbol{y}_d - \boldsymbol{X}_d^T \boldsymbol{w}^{(k)}$$

- For other activate function:

$$y = f_a\left(\sum_{i=1}^{n} w_i x_i\right) = f_a\left(\boldsymbol{x}^T \boldsymbol{w}\right)$$

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \mu \frac{e_k \boldsymbol{x}_d}{\|\boldsymbol{x}_d\|^2}$$

$$e_k = y_d - f_a\left(\boldsymbol{x}_d^T \boldsymbol{w}^{(k)}\right)$$

BACKPROPAGATION ALGORITHM

- We denote the weights for inputs into the hidden layer by w_{ji}^h
- denote the weights for inputs from the hidden layer into the output layer by w_{sj}^o
- input to the j-th neuron in the hidden layer: v_j
- output of the j-th neuron in the hidden layer: z_j

$$
\begin{aligned}
v_j &= \sum_{i=1}^{n} w_{ji}^h x_i, \\
z_j &= f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)
\end{aligned}
$$

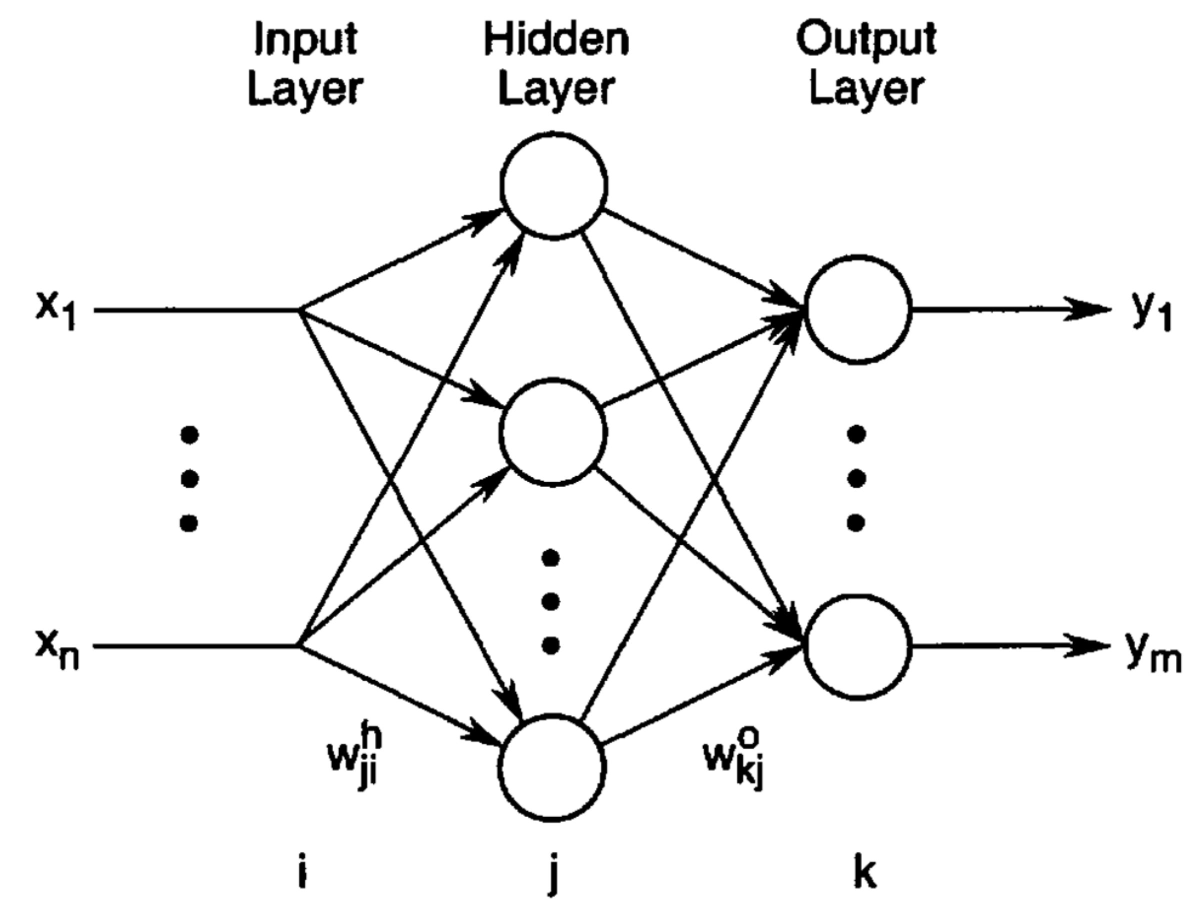


Figure 13.6 A three-layered neural network

BACKPROPAGATION ALGORITHM

- The output from the sth neuron of the output layer is:

$$y_s = f_s^o\left(\sum_{j=1}^{l} w_{sj}^o z_j\right)$$

- Therefore, the relationship between the inputs x_i and the outputs y_s is given by

$$\begin{aligned} y_s &= f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h(v_j)\right) \\ &= f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)\right) \\ &= F_s(x_1, \ldots, x_n). \end{aligned}$$

- The overall mapping that the neural network implements is therefore given by:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} F_1(x_1, \ldots, x_n) \\ \vdots \\ F_m(x_1, \ldots, x_n) \end{bmatrix}$$

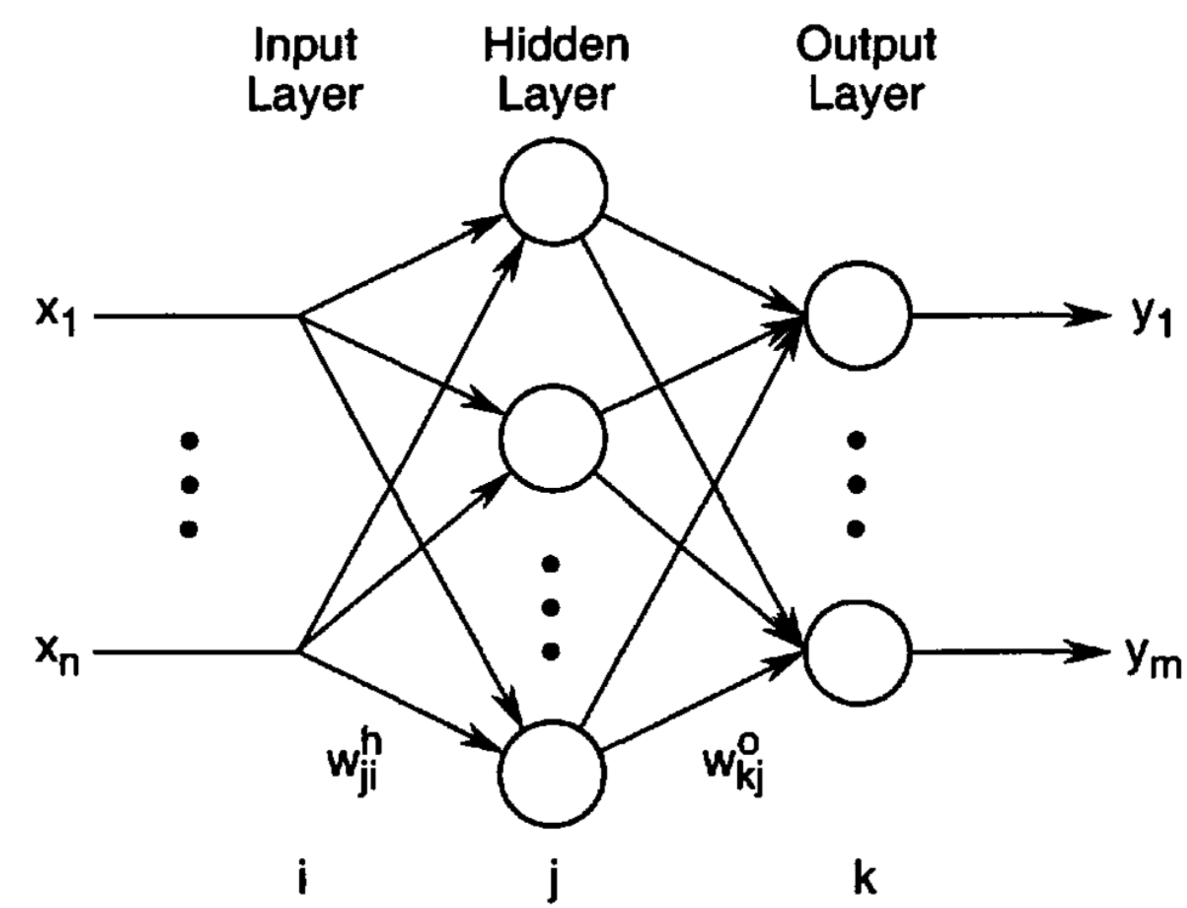


Figure 13.6 A three-layered neural network

BACKPROPAGATION ALGORITHM

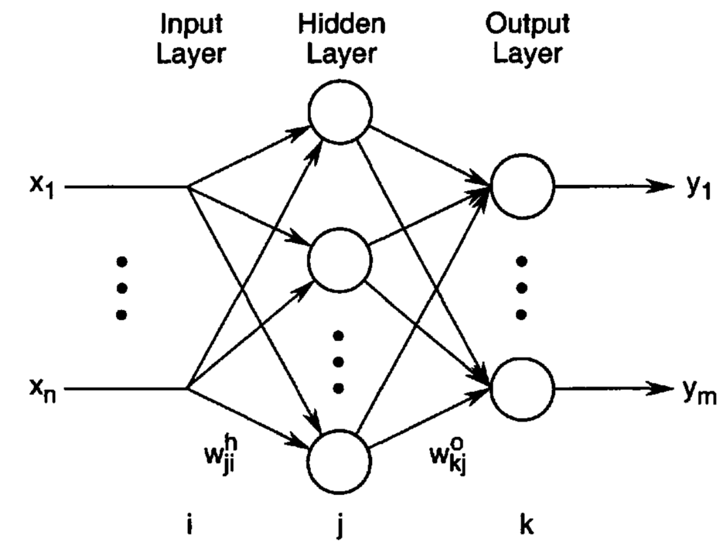


Figure 13.6 A three-layered neural network

- Optimization Problem:

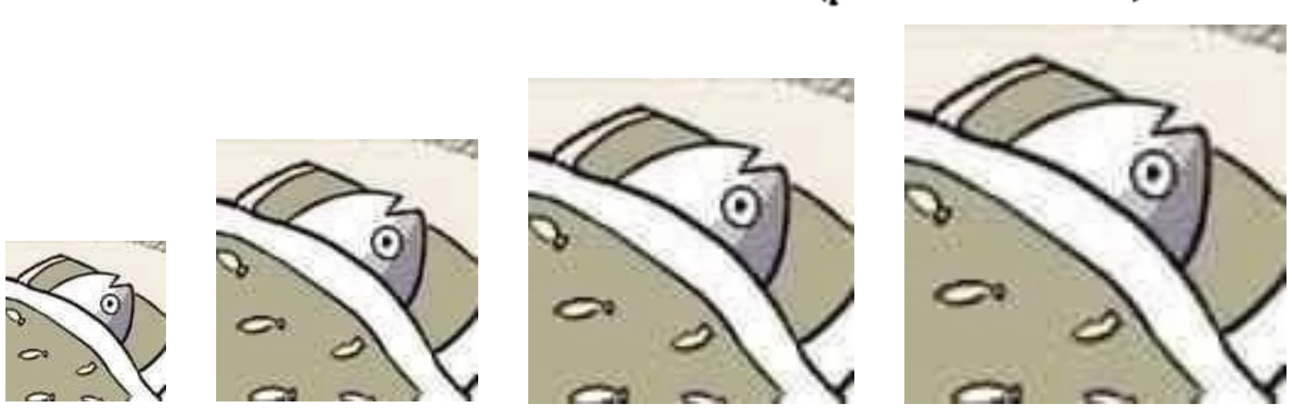
$$\text{minimize } \frac{1}{2}\sum_{s=1}^{m}(y_{ds}-y_s)^2$$

$$y_s = f_s^o\left(\sum_{j=1}^{l} w_{sj}^o f_j^h\left(\sum_{i=1}^{n} w_{ji}^h x_i\right)\right)$$

- 经过艰难的求导(p226)，得到递推公式：

$$\begin{aligned} w_{sj}^{o(k+1)} &= w_{sj}^{o(k)} + \eta\delta_s^{(k)} z_j^{(k)} \\ w_{ji}^{h(k+1)} &= w_{ji}^{h(k)} + \eta\left(\sum_{p=1}^{m}\delta_p^{(k)} w_{pj}^{o(k)}\right) f_j^{h'}(v_j^{(k)}) x_{di}, \end{aligned}$$

$$\begin{aligned} v_j^{(k)} &= \sum_{i=1}^{n} w_{ji}^{h(k)} x_{di}, \\ z_j^{(k)} &= f_j^h\left(v_j^{(k)}\right), \\ y_s^{(k)} &= f_s^o\left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)}\right), \\ \delta_s^{(k)} &= \left(y_{ds} - y_s^{(k)}\right) f_s^{o'}\left(\sum_{q=1}^{l} w_{sq}^{o(k)} z_q^{(k)}\right) \end{aligned}$$



Forward pass: compute the quantities $v_j^{(k)}$, $z_j^{(k)}$, $y_s^{(k)}$ and $\delta_s^{(k)}$

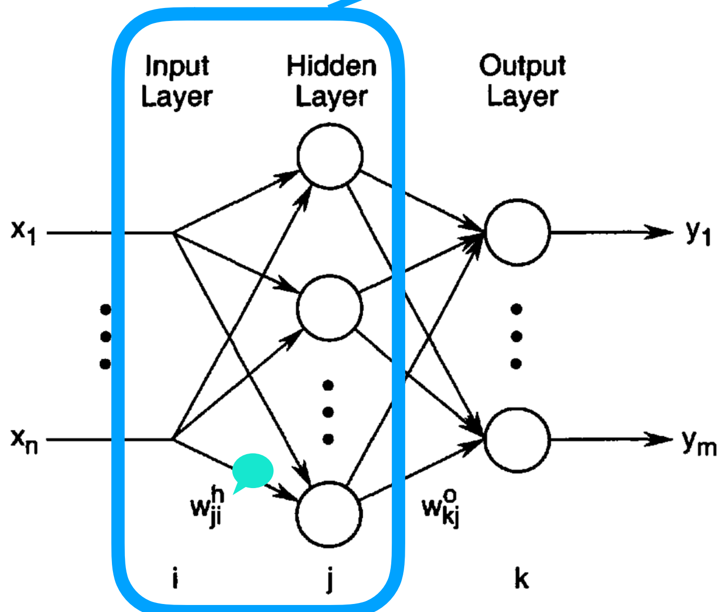


Figure 13.6 A three-layered neural network

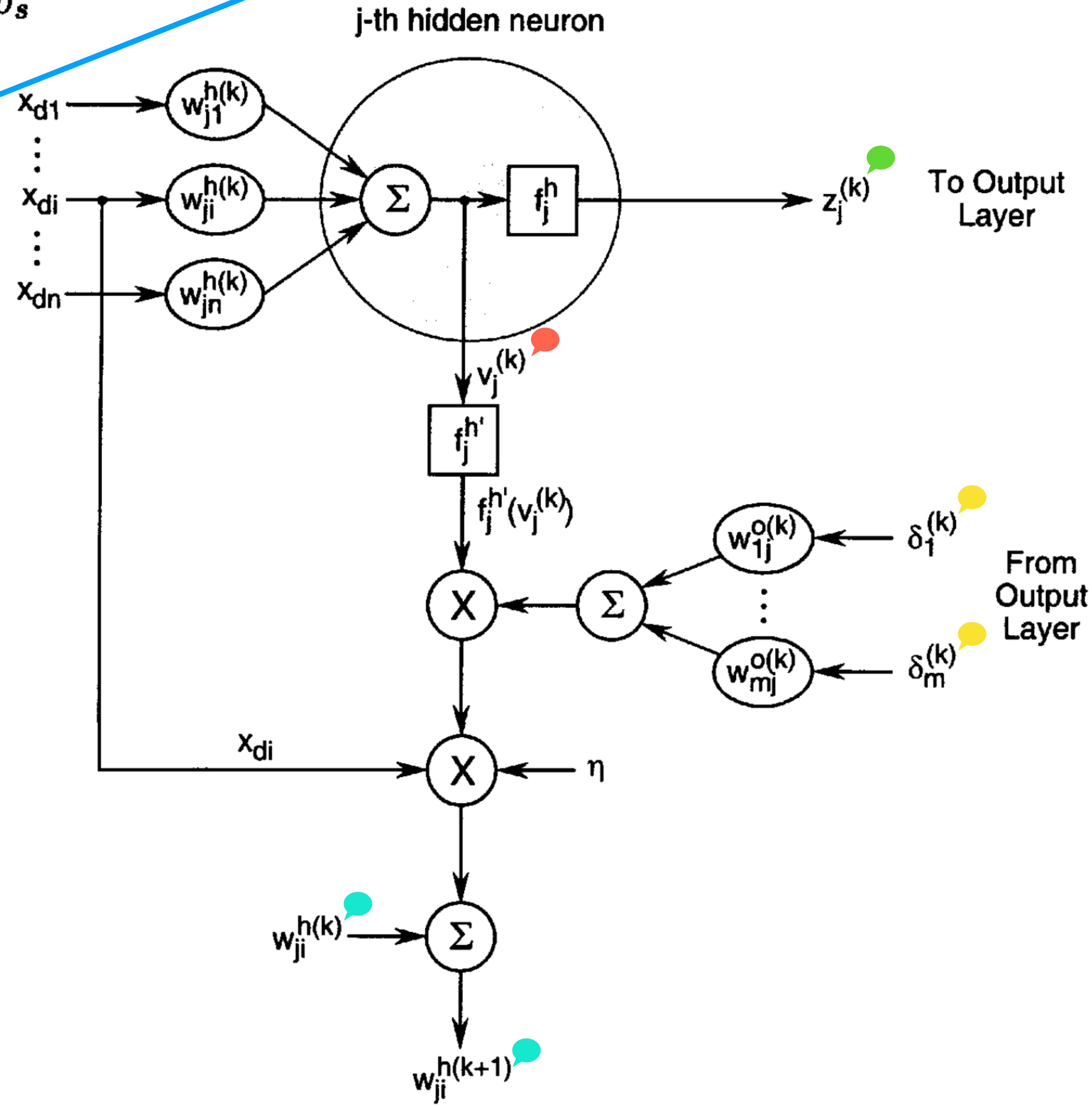


Figure 13.8 Illustration of the update equation for the hidden layer weights

Reverse pass: involves propagating the computed output errors $\delta_s^{(k)}$ backward

Forward pass: compute the quantities $v_j^{(k)}$, $z_j^{(k)}$, $y_s^{(k)}$ and $\delta_s^{(k)}$

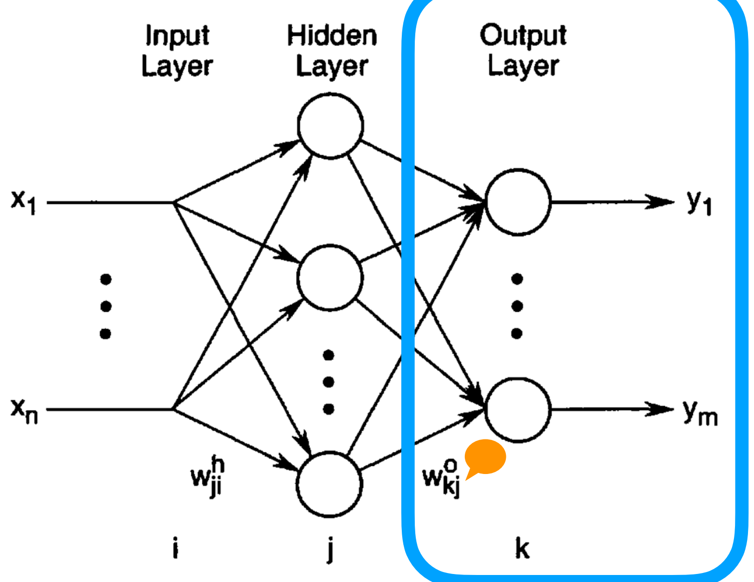


Figure 13.6 A three-layered neural network

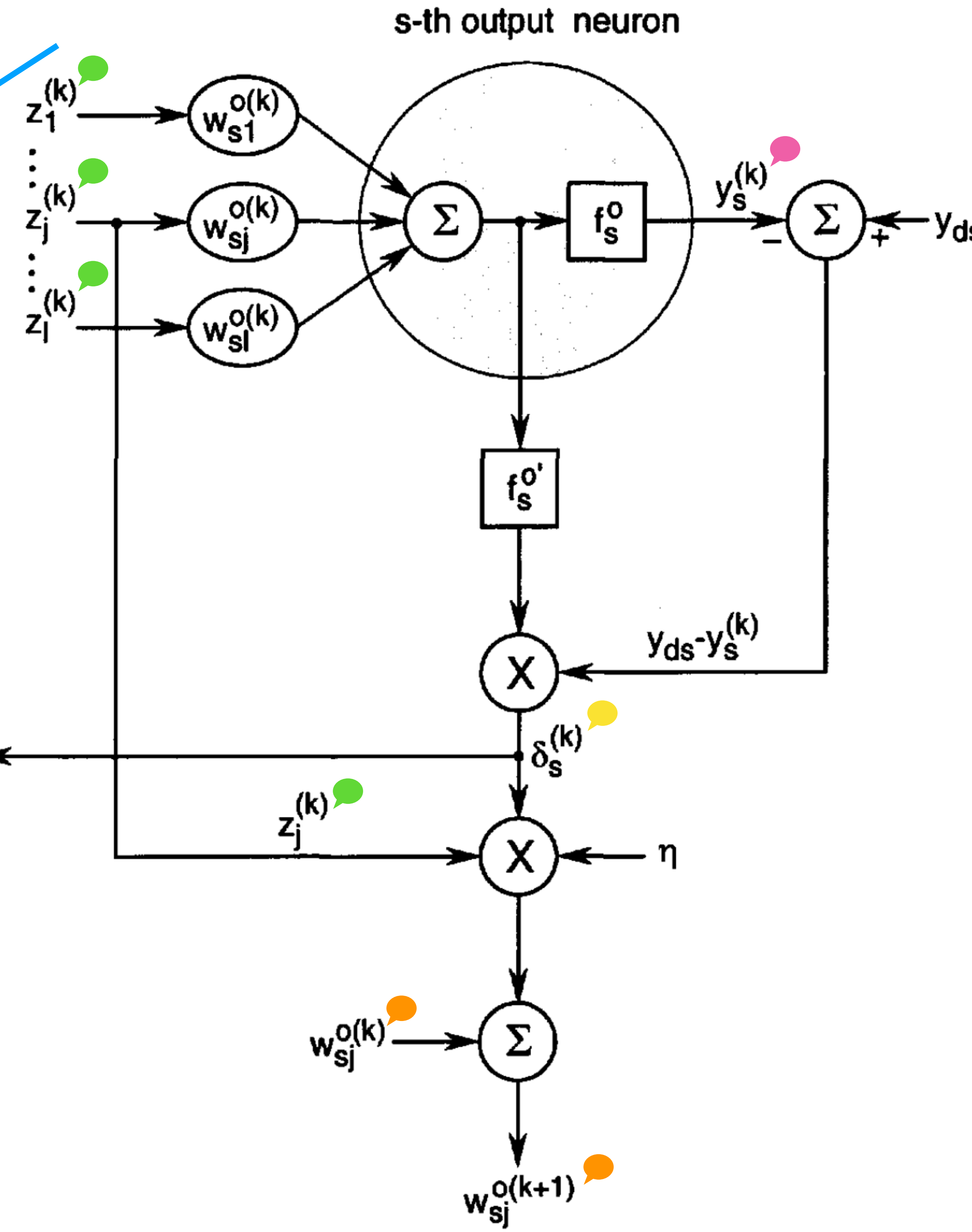


Figure 13.7 Illustration of the update equation for the output layer weights

Reverse pass: involves propagating the computed output errors $\delta_s^{(k)}$ backward

Sigmoid Function

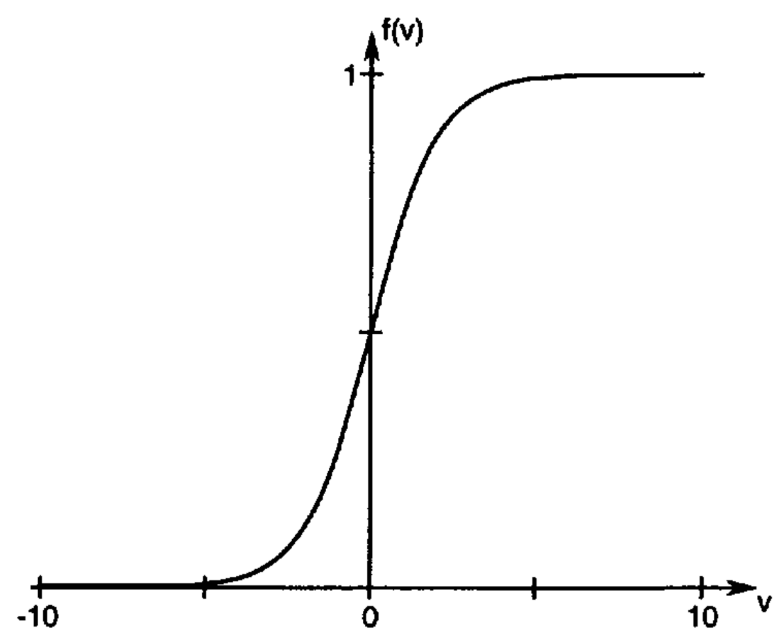


Figure 13.10 The sigmoid function

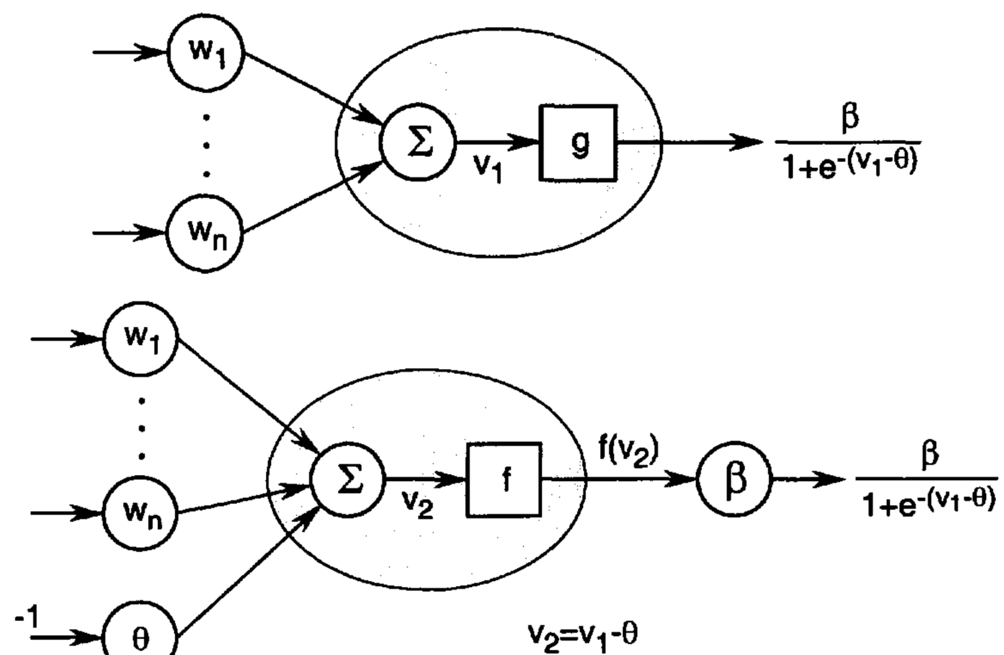


Figure 13.11 The above two configurations are equivalent

$$f(v) = \frac{1}{1+e^{-v}}$$

general version $$g(v) = \frac{\beta}{1+e^{-(v-\theta)}}$$

The parameters β and θ represent *scale* and *shift* parameters

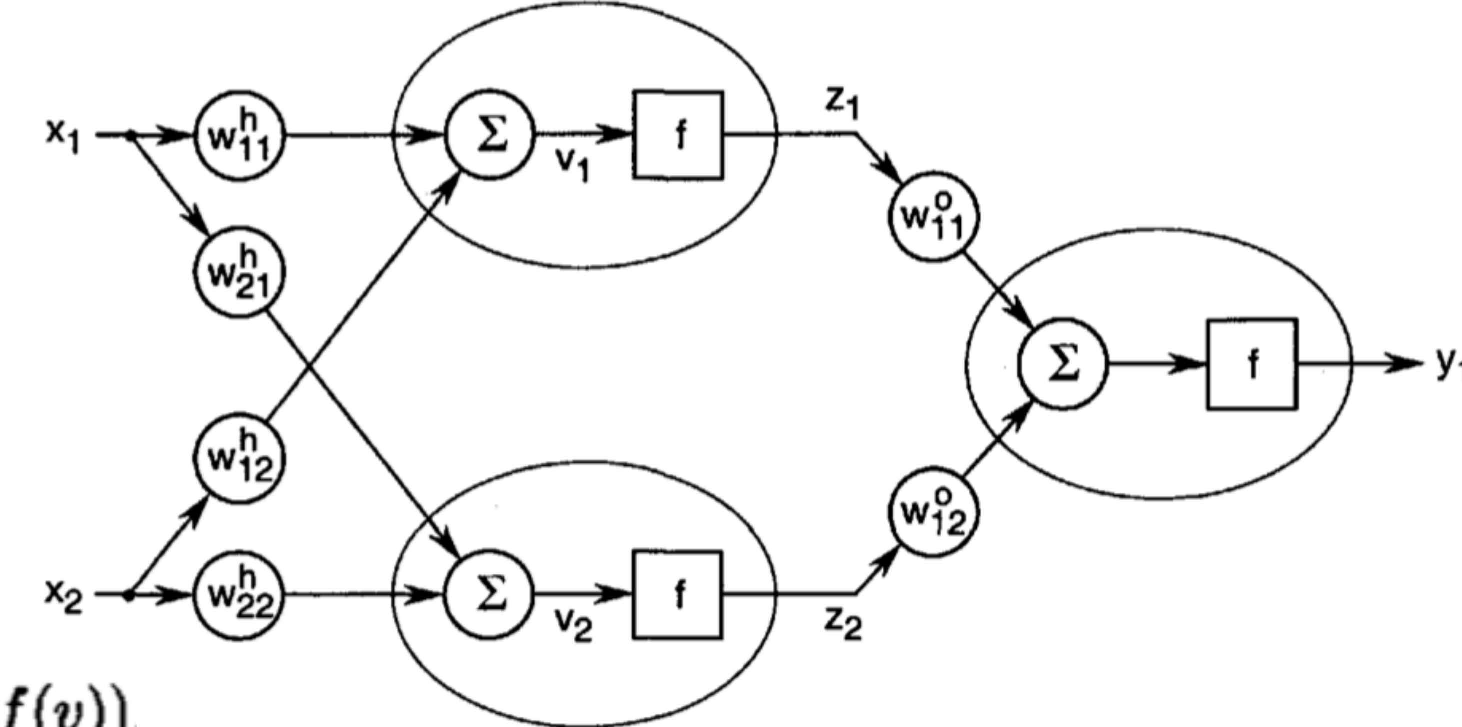
parameter θ is often interpreted as a threshold

Treating them as additional weights in the backpropagation algorithm. Specifically, we can represent a neuron with activation function *g* as one with activation function f with the addition of two extra weights

Exercise

- 这是一道填空题。不许看书！

- Active Function:



$$f(v) = 1/(1+e^{-v}) \quad f'(v) = f(v)(1-f(v))$$

$$w_{11}^{h(0)} = 0.1 \qquad \text{Let } \boldsymbol{x}_d = [0.2, 0.6]^T$$

$$w_{12}^{h(0)} = 0.3 \qquad w_{11}^{o(0)} = 0.4$$

$$w_{21}^{h(0)} = 0.3 \qquad w_{12}^{o(0)} = 0.6$$

$$w_{22}^{h(0)} = 0.4 \qquad y_d = 0.7 \quad \eta = 10$$

$$\delta_1 = (y_d - y_1) f'\left(\sum_{q=1}^{2} w_{1q}^{o} z_q\right)$$

$= (y_d - y_1)$ (1)写出表达式

$$= (y_d - y_1) y_1 (1 - y_1).$$

$v_1^{(0)} = w_{11}^{h(0)} x_{d1} + w_{12}^{h(0)} x_{d2} =$ (2)一位小数

$$v_2^{(0)} = w_{21}^{h(0)} x_{d1} + w_{22}^{h(0)} x_{d2} = 0.3$$

$z_1^{(0)} = f(v_1^{(0)}) =$ (3)表达式 $= 0.5498$

$$z_2^{(0)} = f(v_2^{(0)}) = \frac{1}{1+e^{-0.3}} = 0.5744.$$

$y_1^{(0)} = f\left(w_{11}^{o(0)} z_1^{(0)} + w_{12}^{o(0)} z_2^{(0)}\right) = f($ (4)四位小数 $) = 0.6375$

$$\delta_1^{(0)} = (y_d - y_1^{(0)}) y_1^{(0)} (1 - y_1^{(0)}) = 0.01444.$$

$w_{11}^{o(1)} =$ (5)表达式 $= 0.4794$

$$w_{12}^{o(1)} = w_{12}^{o(0)} + \eta \delta_1^{(0)} z_2^{(0)} = 0.6830$$

Thank you!